COLLECTION AND ANALYSIS OF ACOUSTIC DOPPLER CURRENT PROFILER DATA

Final Technical Report

by

W R Roy Richardson & Colin R Thorne

The University of Nottingham

September 1995 - January 1996

United States Army

EUROPEAN RESEARCH OFFICE OF THE U.S. ARMY

London England

CONTRACT NUMBER: R & D 7791-EN-09

The Research reported in this document has been made possible through the support and sponsorship of the U.S. Government through its European Research Office of the U.S. Army. This report is intended for the internal management use of the University of Nottingham and U.S. Government.

SUMMARY

Acoustic Doppler Current Profilers are presently employed by the U.S. Army in a number of projects. However, as with most new techniques there have been several problems both with the collection and the subsequent analysis of data. The main objective of this work is to improve the accuracy of Acoustic Doppler Current Profilers by examining the existing survey techniques and by developing computer software to process survey data. Two FORTRAN programs have been developed during the contract period. One program is designed to combine data from an external differential global positioning system with data from an ADCP to give a more reliable discharge measurement, and the second program is designed to in-fill gaps in records due to missing ADCP velocity data. In order to test and develop the software, data was collected on the Columbia River, Washington. This document details the resulting software designed and briefly reports on the fieldwork undertaken.

CONTENTS

I. CONVERSION FACTORS FOR MEASUREMENT UNITS

Because of the parties involved in this research project, namely The University of Nottingham, UK, and the United States Government, a table of conversion factors for Imperial and SI units is given below:

FROM	*TO*	MULTIPLY BY
Inches (in)	*Millimetres (mm)*	25.4
Feet (ft)	*Metres (m)*	0.3048
Square Feet (ft^2)	*Square Metres (m^2)*	0.0929
Cubic Feet (ft^3)	*Cubic Metres (m^3)*	0.02832
Yards (yd)	*Metres (m)*	0.9144
Miles (Mi)	*Kilometres (km)*	1.609
Millimetres (mm)	*Inches (in)*	0.0394
Metres (m)	*Feet (ft)*	3.281
Square Metres (m^2)	*Square Feet (ft^2)*	10.7639
Cubic Metres (m^3)	*Cubic Feet (ft^3)*	35.3147
Metres (m)	*Yards (yd)*	1.0936
Kilometres (km)	*Miles (Mi)*	0.621

II. LIST OF SYMBOLS

E	Corrected Easting
N	Corrected Northing
e	GPS Easting
n	GPS Northing
m	Survey line gradient
c	Survey line intercept
β	Local magnetic declination
ρ	True ADCP direction
ψ	Measured ADCP direction
V_W	Absolute water velocity
V_B	Boat velocity
V_{ADCP}	ADCP measured water velocity
R	ADCP range
D_n	Ensemble depth
D_{ADCP}	ADCP transducer depth
θ	ADCP transducer beam angle
U	Velocity at height Z above bed
U^*	Shear velocity
Z_O	Bottom roughness height
b	Power Law exponent
a'	Non-ADCP parameters in the power law equation
Q_{top}	Top layer discharge
W	Corrected ensemble width
D_{blank}	Blank depth
Q_{bin}	Bin discharge
V	Corrected velocity magnitude
α	Survey line direction
H	Bin size
Q_{btm}	Bottom layer discharge
Q_{shore}	Near shore discharge
V_m	Nearest ensemble depth averaged velocity
L	Distance to bank
D_m	Nearest ensemble depth
X_i	Vector cross product
h_i	Bin height
Δt	Time elapsed during ensemble
E_{boat}	East velocity of the survey vessel
N_{boat}	North velocity of the survey vessel
E_{water}	East velocity of the water
N_{water}	North velocity of the water

III. LIST OF FIGURES

1. INTRODUCTION

Acoustic Doppler Current Profilers (ADCPs) are "state-of-the-art" instruments that measure water velocities acoustically, in particular the Doppler effect. The Doppler effect is the change in observed sound pitch that results from relative motion between a source and observer. The major advantages of ADCPs over more traditional methods of river survey are their speed of use, quality of data, resolution of measurement and their ability to be used onboard moving vessels to measure river discharges (Didden, 1989)

However, due to their nature of operation ADCPs produce large quantities of data that requires extensive post-survey processing in order to be converted into a usable format. Additionally when being used onboard a moving vessel and under certain conditions (i.e. high flows) their ability to "bottom-track" breaks down (Münchow et al., 1995). The ADCP records the velocity of the channel bed as well as the water velocity and makes a correction for vessel motion during measurement. In order to overcome this "bottom-track" problem new survey techniques and post-survey data processing software have been designed. The "bottom-track" facility is replaced by an external Differential Global Positioning System (DGPS) to monitor and record vessel motion.

During this study test data sets were collected using an ADCP in conjunction with a DGPS to measure water velocities and river discharge on the Columbia River, upstream and downstream of the Bonneville Dam, Washington. GPSQ a DOS based FORTRAN computer software package was then developed to combine the data files from the ADCP with the files recorded from the DGPS.

In addition to the "bottom-track" problem, ADCP data can be prone to missing ensembles in a given cross-section. These missing data give rise to accumulated errors in discharge and

positioning data when the ADCP is being used in a standard mode (i.e. without an external DGPS). During this study a second DOS based FORTRAN program (ADCPFILL) was developed to fill-in missing data.

2. PROJECT CHRONOGRAPHY

2.1 ITINERARY

1ST September 1995:

Mr Richardson and Prof. Thorne arrived in Vicksburg, Mississippi. After initial meetings Mr Richardson began work at Waterways and met with Mike Trawle, Tim Fagerburg and Thad Pratt to outline possible areas of concern with present ADCP data collection techniques and analysis.

2nd September to 30th September 1995:

Bottom-track errors were highlighted as a major problem in the ADCP river gauging technique, and subsequently Mr Richardson began developing a method for calculating discharge using a Differential Global Positioning System to replace bottom-track. He also started designing general code for GPSQ software.

1st October to 14th October 1995:

Field work in Washington, on Columbia River. Mr Richardson and Thad Pratt collected sample data sets necessary for software development and refined survey methods for collecting and logging DGPS data.

16th October to 1st November 1995:

Mr Richardson and Thad Pratt returned to Vicksburg to refine and complete software development for GPSQ. Problems of missing ADCP data in all ADCP applications became apparent - ADCPFILL was designed to overcome this.

2nd November 1995:

Mr Richardson returned to U.K.

2.2 DATA COLLECTION

In order to overcome the inherent bottom-track problem within the ADCP setup, an alternative to bottom-tracking must be utilised to monitor boat motion. An ideal alternative is a Differential Global Positioning System (DGPS). A DGPS achieves reasonable accuracy in absolute position and it is also capable of greater precision when monitoring relative positions or velocity (FAP24[1], 1993).

A trial data set was collected on the Columbia River, Washington upstream and downstream of the Bonneville Dam. Data was collected using an RDI Instruments ADCP with an acoustic frequency of 600 kHz in conjunction with a Trimble Navigations DGPS. A test data set was needed for a number of reasons. First, it was necessary in order to develop the software required for the subsequent corrections applied to the raw ADCP data. Second, the technique of using a DGPS in conjunction with an ADCP for gauging discharge required verification. Finally, a survey method for setting up and logging data from the DGPS was required.

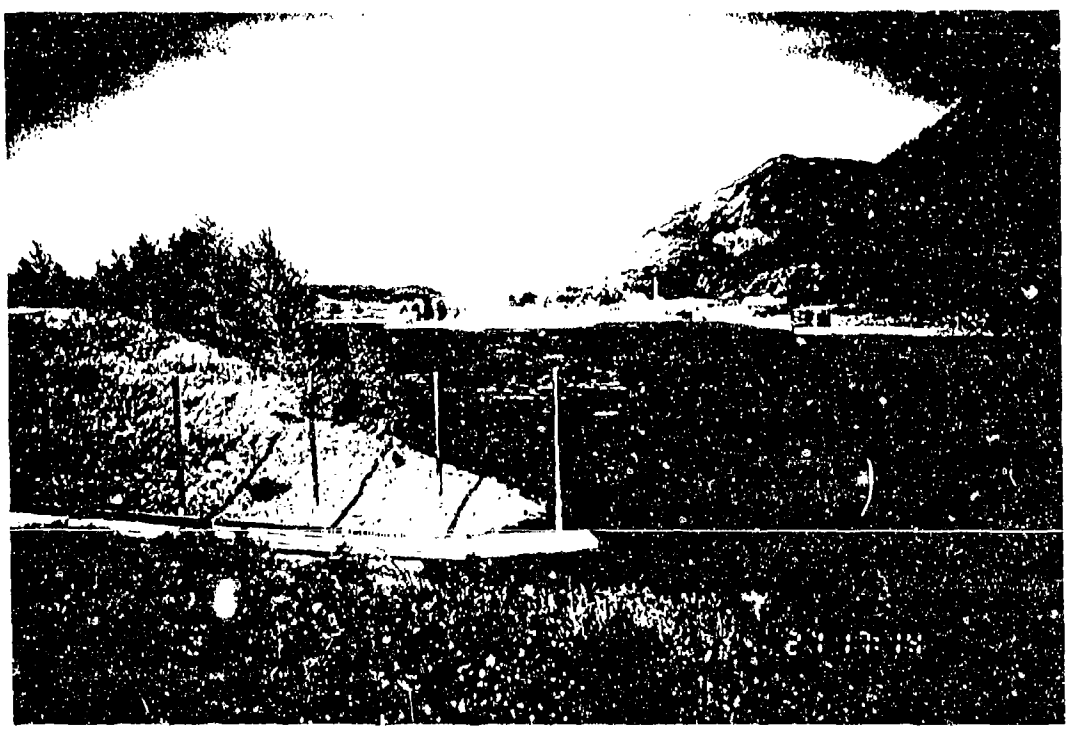


Plate 1. Test Site on Columbia River, Washington

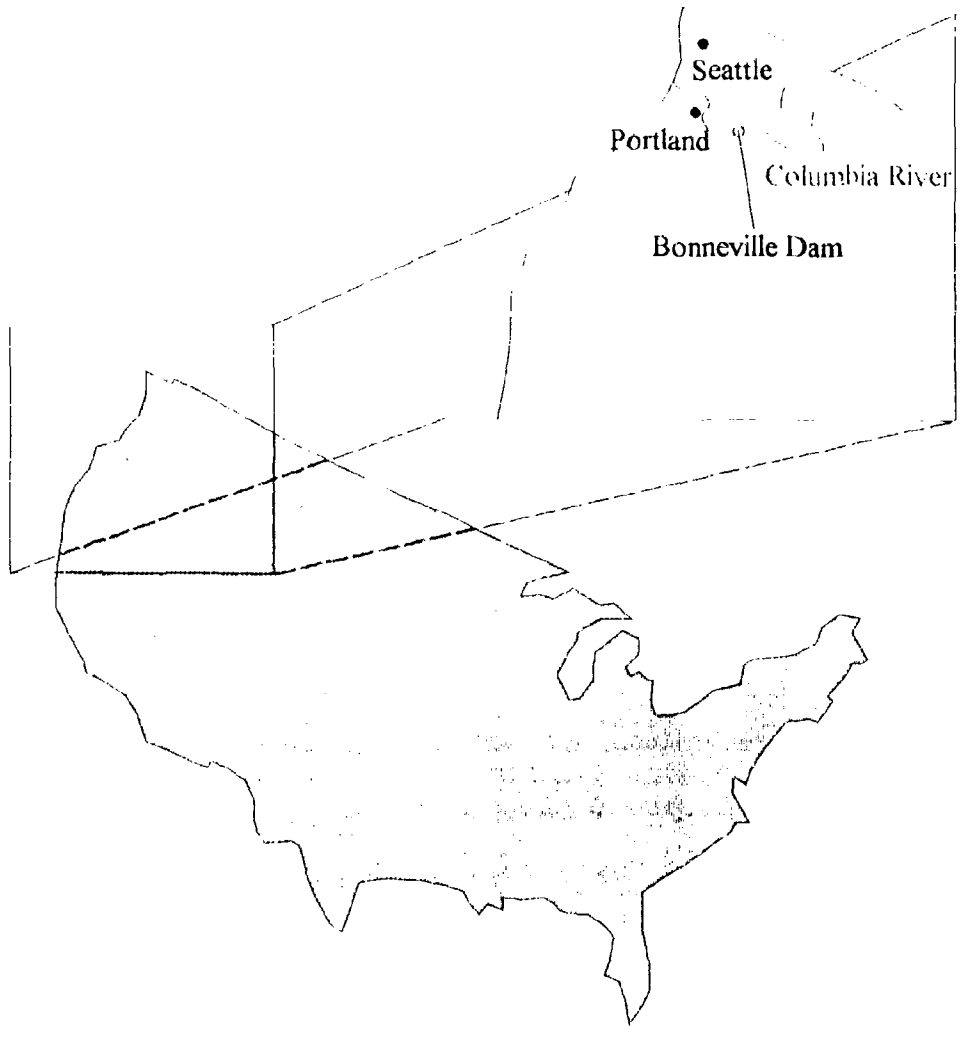


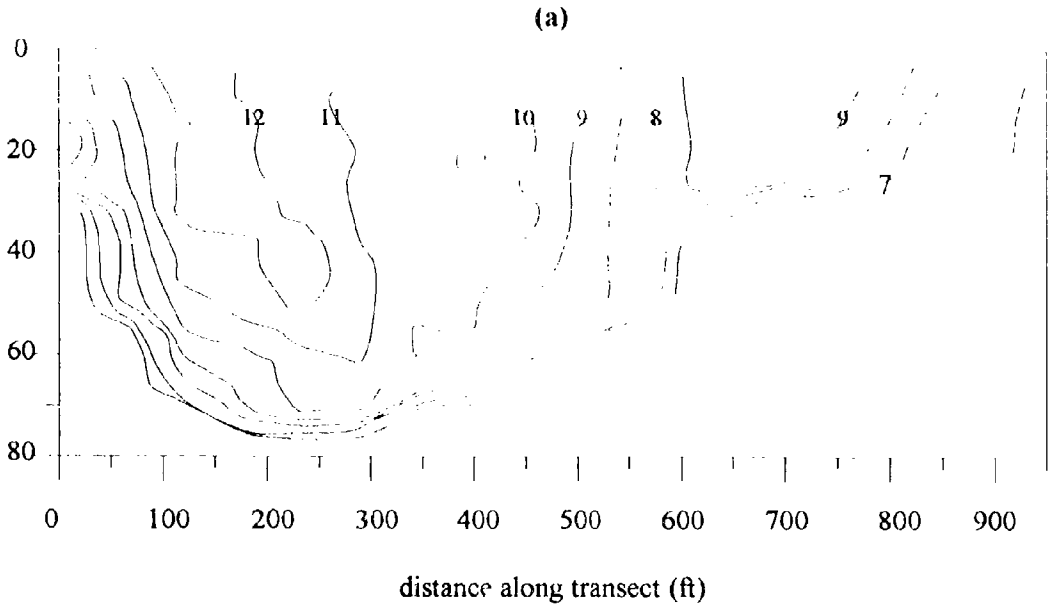
Figure 1. Location Map of Test Data Site, Bonneville Dam, Columbia River, Washington.

2.3 RESULTS

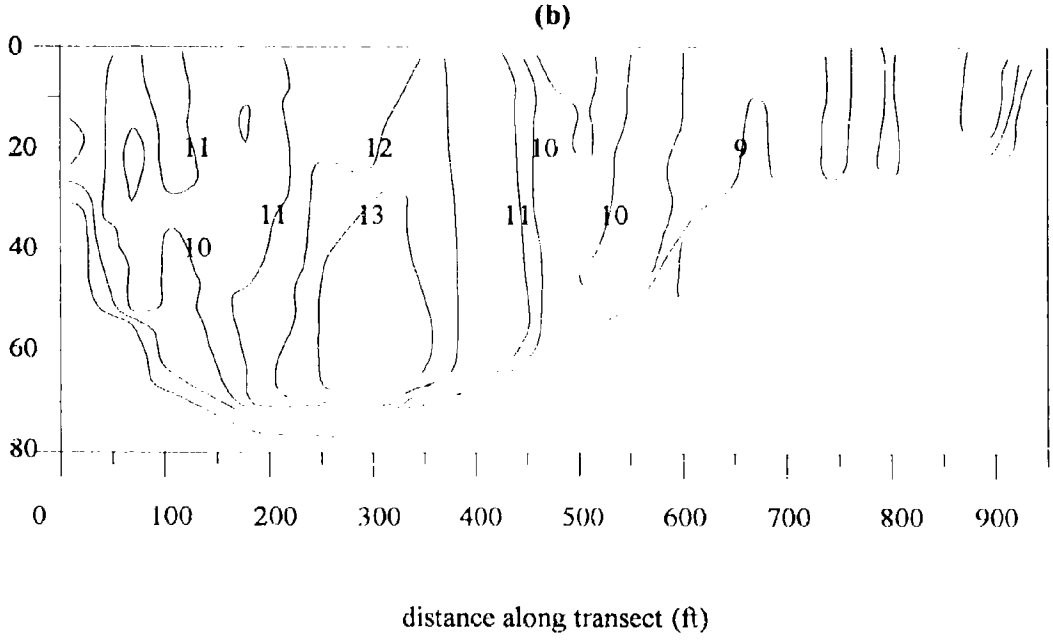
Figure 1.2 shows results obtained at one cross-section during the test survey downstream of Bonneville Dam. Figure 1.2(a) shows results from the ADCP used in bottom-track mode, with a discharge calculated of 130143 cfs. Figure 1.2(b) shows results obtained using the ADCP in conjunction with the DGPS, with a discharge calculated using 'GPSQ' of 191750 cfs. It is obvious from the plots of measured velocities in figure 1.2 that the velocity field shown is significantly

different dependent on the velocity reference used. Also, the discharge calculated using the DGPS as a velocity reference is 47% greater than that when using bottom-track. The 'bottom-track declination' calculated by 'GPSQ' at the time of data processing is 21 degrees (the bottom-track declination is the difference between the survey line direction calculated using bottom track data and the survey line direction calculated using the DGPS data).

The discharge results may be explained in part by the theory that bottom-track errors lead to an under-estimation of the water velocities and therefore discharge, through an under-estimation of vessel velocity. The bottom-track declination of 21 degrees may be partly due to very localised magnetic affects on board the survey vessel interfering with the internal ADCP compass. This affects the measurement of velocities in two ways. First, the bottom track measurement of the vessel's speed and direction is affected. Second, the measurement of water velocities made relative to the ADCP is affected. When these two measurements are combined to calculate an absolute water velocity, the affect of local magnetic declination is compounded. The DGPS is not susceptible to the affects of local magnetic declination and can therefore eliminate the affect from the measurement of vessel velocity.



discharge: 130143 cfs



discharge: 191750 (cfs)

Figure 1.2 Measured velocities (ft/s); (a) 'Bottom-track' referenced and (b) GPS referenced

3. GPSQ

3.1 INTRODUCTION

When ADCPs are used on board moving vessels for river discharge measurements, they are generally configured to record the velocity of the bed (bottom track) and the velocity of the water relative to the ADCP. The ADCP then combines the two velocities to give an absolute water velocity. Under certain conditions ADCP bottom track data can become unreliable and thus introduce errors into the measurement of water velocities and discharge. ADCP bottom track is limited by the convex arrangement of the ADCP transducer heads, making it impossible for the ADCP to measure the entire water column. When the ADCP is measuring what it assumes to be the bottom, it is in fact measuring a moving surface of wat /sediment just above the bed. Under normal conditions the echo from the hard bottom surface is enough to overcome this problem, but under higher flow conditions, when sediment is being entrained into the water and the bed is mobile, errors are introduced. Generally, bottom track errors of this nature will lead to an under-estimation of the discharge due to the ADCP using a velocity reference that is moving in the downstream direction.

'GPSQ' is a DOS based program written in FORTRAN which, together with an amended survey technique, will re-calculate ADCP measured velocities and discharge. It does this by utilising data from an external differential global positioning system (DGPS), to replace bottom track and measure boat motion. GPSQ uses ADCP velocity data that is taken from the RDI-TRANSECT program, in the usual ASCII processed data format, with the velocity reference set to "none". DGPS files are then combined with the ADCP ASCII files to correct for boat motion and produce correct velocities and discharge.

3.2 METHOD

GPSQ works in three modules. The program modules are:

(1) DGPS data processing module

(2) Velocity data correcting module

(3) Discharge calculating module

3.2.1 DGPS Data Processing

The first module of GPSQ reads the GPS navigation file and calculates the boat motion and position for each ADCP ensemble. The module will read GPS data in the following format;

```
ENSEMBLE 1038 PCTIME 4685248
10:01:56 1624658.61 718868.14
10:01:57 1624656.07 718866.96
10:01:58 1624654.36 718866.70
10:01:59 1624652.71 718867.11

ENSEMBLE 1039 PCTIME 4685605
10:02:00 1624651.46 718868.14
10:02:01 1624650.64 718869.64
10:02:02 1624649.27 718871.14
10:02:03 1624648.40 718873.53
10:02:04 1624648.40 718873.53

ENSEMBLE 1040 PCTIME 4686034
10:02:05 1624647.33 718876.20
10:02:06 1624646.06 718878.92
10:02:07 1624644.58 718881.78
10:02:08 1624642.61 718885.42
```

The ensemble header line gives an ensemble number followed by an ensemble start time (PCTIME) in hundredths of a second. The GPS data lines give a position time (hours:minutes:seconds) followed by an Easting and Northing. The GPS data processing module reads this file into an array of ensemble number with PCTIME from the end of the ensemble to

the start of the transect, and an array of GPS East and North with time from the start of the first ensemble. The two arrays are then matched in terms of time to give each ensemble number a position at the end of the ensemble and an average boat velocity during the ensemble (a velocity is calculated for each GPS record and then averaged over the ensemble period).

The next segment of this module then calculates a survey line definition based on the GPS data. The start and end points of the GPS record are used as the start and end points for the survey line. A definition of the survey line is necessary for the discharge calculation method described below. A distance along the survey line (course made good; CMG) and a width is calculated for each ensemble, together with a corrected position. GPS positions are then translated to survey line positions using the following equations:

$$E = (m.n + e - c.m) / (m^2 + 1) \quad \text{(Eqn 3.1)}$$

$$N = m.E + c \quad \text{(Eqn 3.2)}$$

where:

E	=	Corrected Easting
N	=	Corrected Northing
e	=	GPS Easting
n	=	GPS Northing
m	=	Survey line gradient (dist. North / dist. East)
c	=	Survey line intercept

The output of the GPS processing module is in the format shown below, Velocities and course made good data are given in the selected data output units, US customary or SI. The GPS processed data file is a temporary file which is removed at the end of the program execution.

ENS	CMG	EASTING	NORTHING	E VEL	N VEL	WIDTH
1038	2.96	1624658.00	718868.90	5.81	-5.33	1.43
1039	18.76	1624656.00	718873.20	25.93	-3.69	5.67
1040	52.84	1624651.00	718882.40	42.18	-5.20	9.05
1041	98.75	1624645.00	718894.80	49.46	-6.25	9.54
1042	155.82	1624636.00	718910.10	64.82	-7.35	9.78
1043	206.34	1624629.00	718923.70	36.06	-7.48	9.68
1044	252.05	1624623.00	718936.00	55.45	-5.97	10.25
1045	313.51	1624614.00	718952.60	67.55	-5.54	11.29

3.2.2 Velocity Correction

The next module in GPSQ takes the processed GPS data and combines it with the processed ADCP ASCII data, to give corrected water velocities. The ADCP processed ASCII files are in the format shown in Appendix A.

For each measurement bin the measured velocity direction is corrected for the local magnetic declination (β) as input by the user.

$$\beta = \rho - \psi \quad \text{(Eqn 3.3)}$$

where:

β = Local magnetic declination

ρ = True ADCP direction

ψ = Measured ADCP direction

The non-referenced East and North ADCP velocities and then corrected for boat motion to give an absolute water velocity.

$$V_W = V_B + V_{ADCP} \quad \text{(Eqn 3.4)}$$

where:

V_W = Absolute water velocity

V_B = Boat velocity given by DGPS

V_{ADCP} = ADCP non-referenced measured water velocity

A depth is calculated for each ensemble using either ADCP beam depth data given in the ADCP ASCII file, or external fathometer depth data in a format as yet untested in the field. The ensemble depth is used to calculate an ADCP range. Velocity data outside of this range is discarded. The range of acceptable ADCP data is given by the following equation:

$$R = (D_n - D_{ADCP}) \cdot \cos\theta \quad \text{(Eqn 3.5)}$$

where:

R = ADCP range

D_n = Ensemble depth

D_{ADCP} = ADCP transducer depth (given in ADCP ASCII file)

θ = ADCP transducer beam angle (INPUT by user)

The output from the velocity data correction module is given in the file VELOCITY.DAT which is created in the working directory. VELOCITY.DAT is in the following format:

id, ens,	dist,	east,	north,	width,	depth,	bin,	mag,	dir,	east,	north,	vert,	error,	db avg,	adcp range
1 1038	2.97	1624658.00	718868.90	5.81	24.92	5.00	5.61	297.30	-4.99	2.57	.00	-1.70	247.25	23.48
1 1038	2.97	1624658.00	718868.90	5.81	24.92	6.70	5.49	288.80	-5.20	1.77	-.10	-.60	235.50	23.48
1 1038	2.97	1624658.00	718868.90	5.81	24.92	8.30	5.19	282.71	-5.06	1.14	.00	-.40	206.25	23.48
1 1038	2.97	1624658.00	718868.90	5.81	24.92	9.90	5.37	277.87	-5.32	.74	.20	-.60	180.25	23.48
1 1038	2.97	1624658.00	718868.90	5.81	24.92	11.60	4.52	282.69	-4.41	.99	.10	-.10	172.50	23.48
1 1038	2.97	1624658.00	718868.90	5.81	24.92	13.20	5.25	291.11	-4.90	1.89	.20	-.20	163.00	23.48
1 1038	2.97	1624658.00	718868.90	5.81	24.92	14.90	5.46	290.98	-5.10	1.95	.20	-.70	159.25	23.48
1 1038	2.97	1624658.00	718868.90	5.81	24.92	16.50	5.65	297.86	-4.99	2.64	.10	-.90	153.50	23.48
1 1038	2.97	1624658.00	718868.90	5.81	24.92	18.10	5.86	303.79	-4.87	3.26	-.20	.00	150.50	23.48
1 1038	2.97	1624658.00	718868.90	5.81	24.92	19.80	5.14	289.47	-4.85	1.71	-.10	-.10	165.50	23.48
2 1039	18.77	1624656.00	718873.20	25.94	25.50	5.00	5.25	307.51	-4.17	3.20	.20	-.10	211.00	24.02
2 1039	18.77	1624656.00	718873.20	25.94	25.50	6.70	5.42	304.90	-4.45	3.10	.10	.10	204.75	24.02

Any bad bins of measurement within the ADCP range are ignored and are not recorded in the corrected velocity file.

3.2.3 Discharge Calculation

A discharge is calculated based on the survey line defined by the GPS data. Each ensemble is assigned a width and position along the survey line and a discharge is calculated through the ensemble. For each ensemble a top layer discharge (estimated), middle layer discharge (measured) and a bottom layer discharge (estimated) are calculated. Two options are available for the top layer discharge estimation, the power or constant method. The missing bottom layer discharge is calculated using the power method.

TOP LAYER DISCHARGE ESTIMATION USING THE POWER METHOD

The power law equation for describing velocity profiles in open channels (Chen, 1991; Simpson & Oltmann, 1990) is given as:

$$U / U^* = 9.5 \cdot (Z / Z_0)^b \quad \text{(Eqn 3.6)}$$

where:

U	=	Velocity at height Z above the bed
U*	=	Shear velocity
Z	=	Height above channel bed
Z_0	=	Bottom roughness height
b	=	Power Law exponent

which can be re-arranged to give:

$$U = (9.5 \cdot U^* / Z_0^b) \cdot Z^b \quad \text{(Eqn 3.7)}$$

$$a' = (9.5 \cdot U^* / Z_0^b) \quad \text{(Eqn 3.8)}$$

where a' represents the non-ADCP parameters, then

$$U = a' \cdot Z^b \quad \text{(Eqn 3.9)}$$

For each measurement bin, GPSQ solves equation 3.9 to give a value of a'. Within each ensemble the a' values are averaged (a'_avg), and the average is then used to solve for the missing top layer velocity. The top layer discharge is then given by;

$$Q_{top} = a'\_avg \cdot Z^b \cdot W \cdot D_{blank} \quad \text{(Eqn 3.10)}$$

where:

Q_{top} = Top layer discharge

W = Corrected ensemble width

D_{blank} = Blank depth

TOP LAYER DISCHARGE ESTIMATION USING THE CONSTANT METHOD

The constant method for estimating the missing discharge near the surface is the simpler of the two techniques. The constant method does not follow accepted hydraulic principles of velocity profile description, however, under certain circumstances it seems to give a more accurate picture of near surface properties. The method simply takes the velocity from the first good measurement bin in the profile (V_{top}) and extrapolates to the surface.

$$Q_{top} = V_{top} \cdot W \cdot D_{blank} \quad \text{(Eqn 3.11)}$$

MIDDLE LAYER MEASURED DISCHARGE

The discharge through each measurement bin is given by the following equation:

$$Q_{bin} = V \cdot \cos(\alpha - \rho) \cdot W \cdot H \quad \text{(Eqn 3.12)}$$

where:

Q_{bin} = Bin discharge

V = Corrected velocity magnitude

α = Survey line direction

ρ = Corrected velocity direction

H = Bin size

The bin size (H) is equal to the distance between the bin being considered and the last good measurement bin. This is usually equal to the configured bin size but will change if any bins are missing due to bad data. The middle layer discharge is then equal to the sum of all measurement bin discharges.

BOTTOM LAYER ESTIMATED DISCHARGE

The missing discharge outside of the ADCP range at the bottom of the velocity profile is calculated using the power law equation (Eqn 2.5). The bottom layer discharge is given by:

$$Q_{btm} = a'\_avg \cdot ((D_n - R) / 2)^b \cdot (D_n - R) \cdot W \quad \text{(Eqn 3.13)}$$

where:

Q_{btm} = Bottom layer discharge

NEAR SHORE DISCHARGE ESTIMATION

GPSQ can be used to estimate any discharge that is not measured close to the banks at the beginning and end of each transect. GPSQ uses the same ratio-interpolation method employed by TRANSECT to estimate near shore discharge. The method uses the first and last known velocities and assumes a tri-angular area between the bank and transect. The missing near shore discharges are given by:

$$Q_{shore} = 0.707 \cdot V_m \cdot L \cdot D_m / 2 \quad \text{(Eqn 3.14)}$$

where:

Q_{shore} = Near shore discharge (start or finish)

V_m = Nearest ensemble depth averaged velocity

L = Distance to bank

D_m = Nearest ensemble depth

The total discharge for the transect is then given by the sum of the top layer discharges, measured discharges, bottom layer discharges and the near shore discharges. The discharge output file, whose name and location is specified by the user, is in the following format:

top layer discharge estimation method: power

power co-efficient 1.667000E-01

ensemble	dist.	easting	northing	width	depth	bin q	top q	btm q	total q
1038	2.97	1624658.00	718868.90	5.81	24.92	-336.10	-95.60	-18.51	-450.21	-2.34
1039	18.77	1624656.00	718873.20	25.94	25.50	-939.54	-263.17	-52.39	-1255.10	-1.43
1040	52.84	1624651.00	718882.40	42.18	30.50	-2363.18	-547.49	-130.89	-3041.56	-1.78
1041	98.74	1624645.00	718894.80	49.46	37.93	-4173.16	688.66	-204.95	-5066.76	-1.83
1042	155.83	1624636.00	718910.10	64.82	44.30	-7750.86	-1012.12	-354.03	-9117.00	-2.00

total discharge -185215.50(cfs)

total cross-sectional area 124070.10(ft^3)

4. ADCPFILL

4.1 INTRODUCTION

ADCPFILL is a DOS based program which will read processed ADCP files from the RDI TRANSECT program and write over bad ensemble data. Bad ensemble data will occur when two or more ADCP transducer beams do not return good data for any reason. Bad ensemble data affects the overall calculation of discharge and the ability of the ADCP to calculate the position of each ensemble, in terms of distance travelled, displacement east and north and straight line distance to starting point (course made good). Therefore, after one bad ensemble the bottom track and discharge data for every subsequent ensemble has a built-in error, which accumulates after every bad data segment.

4.2 METHOD

During playback of raw ADCP files, an ASCII file can be produced (using the ALT-A command) containing position and velocity/discharge data for each ensemble. This ASCII file contains a header followed by velocity data for each ensemble in the transect. The ensemble header contains the time at the start of the ensemble, bottom track data, depth data for each transducer beam, discharge data and various other survey parameters (see appendix A). The discharge data contained in the ensemble header can be used as a test for bad segments of ensembles. Bad ensembles show no increment in the measured discharge, whereas good ensembles show an increase in the cumulative discharge measurement.

Bin data for the ensemble is then given in the form of depth, velocity magnitude, velocity direction, east velocity, north velocity, vertical velocity, error velocity, echo intensities, percentage good data (based on the number of beams used in the solution of the equation of motion of the water) and bin discharge. For bad bins of measurement, data values of -32768 are

attributed to velocities and 2147483647 are attributed to discharge. Bad measurement bins are the result of either the bin being outside the ADCP range or more than one beam returning a bad signal from that particular range of depths.

Once bad ensembles have been identified, ADCPFILL uses a simple linear interpolation to re-calculate ensemble bottom track data, beam depths, pitch, roll, corrected heading, ADCP temperature, bin velocity, echo intensities and percentage good data. Discharge data is not filled by interpolation, but calculated separately. Ensemble time and number are not affected by bad velocity measurements and, therefore, do not need recalculating. ADCPFILL uses the following equation to interpolate between good ensembles:

$$k_i = k_1(n_2-n_i) + k_2(n_i-n_1) / (n_2-n_1) \quad \text{(Eqn 4.1)}$$

where:

k	=	Parameter under consideration
n	=	Ensemble number
$_i$	=	Bad ensemble
$_1$	=	Last good ensemble before bad data segment
$_2$	=	Next good ensemble after bad data segment

Once the bottom track and velocity data for each bad ensemble has been corrected, the bin and ensemble discharges can then be re-calculated. ADCPFILL uses the same moving boat method of discharge calculation employed by TRANSECT (Simpson and Oltmann, 1990; Gordon, 1989). The discharge for any given measurement bin is calculated as follows:

$$Q_{bin} = X_i \cdot h_i \cdot \Delta t \quad \text{(Eqn 4.2)}$$

where:

Q_{bin} = Bin discharge

X_i = Vector cross product

h_i = Bin height

Δt = Time elapsed during ensemble

The vector cross product (X_i) is given by the following formula:

$$X_i = (E_{boat} \cdot N_{water}) - (N_{boat} \cdot E_{water}) \quad \text{(Eqn 4.3)}$$

where:

E_{boat} = East velocity of the survey vessel

N_{boat} = North velocity of the survey vessel

E_{water} = East velocity of the water

N_{water} = North velocity of the water

The accumulated errors in measured discharge and bottom track data are calculated for each bad segment of ADCP data and then added to subsequent ensembles to correct for accumulated errors.

The final output of ADCPFILL is in the same format as the original ADCP data file, and as such can be used by other software packages designed for ADCP ASCII data. The numbers of any bad ensembles corrected are displayed on the screen at the time of processing.

5. CONCLUSIONS & RECOMMENDATIONS

At conception, this project was designed to address the general problem of ADCP data management and analysis on the Mississippi River, in particular to outline the basis for a usable engineering database of ADCP data. However, under certain flow conditions a more fundamental problem was identified, that of actually collecting good quality data using the ADCP. Using experience gained working in Bangladesh with ADCPs on the Brahmaputra River, a new survey and data processing technique was designed and tested.

In the main body of this report two computer programs are detailed, GPSQ and ADCPFILL, which were developed during the study to overcome two of the basic problems of ADCP data collection, bottom-track errors and blank or missing data. These programs were validated and found to work well with the sample data collected on the Columbia River, however they require further development in order to be more flexible for general use and eventually to be incorporated into a standard method of river discharge measurement.

The significant differences between ADCP velocity and discharge data collected using bottom tracking and DGPS, would seem to pose more questions then they answer. Additional work is needed analysing the data collected during the test survey, in particular looking at the affects of local magnetic declination and the possible incorporation of a gyro compass into the survey method.

Further improvements to the survey methods could be achieved through the addition of an external fathometer gauge to measure cross-sectional bathymetry. This would require further software development and field testing. At present depth data is obtained from the ADCP beams, a fathometer would give a more detailed and accurate measurement, thus improving confidence in the discharge calculation. Further work is also needed on procedures to ensure correct

matching of the differential global positioning system file and the ADCP file, in order to ensure that data taken from the two files are combined accurately in time.

6. REFERENCES

Chen, Cheng-Lung (1991), **"Unified Theory on Power Laws for Flow Resistance"**, *Journal of Hydraulic Engineering*, Vol 117, No. 3, March 1991, pp 371-389

Chereskin, TK, and Harding, J (1993), **"Modelling the performance of an Acoustic Doppler Current Profiler"**, *Journal of Atmospheric Oceanic Technology*, Vol. 10, pp 42-63.

Didden, N (1989), **"Performance evaluation of a ship-board 115-kHz ADCP"**, *Continental Shelf Research*, Vol. 7, pp 1232-1243

Gordon, R. L. (1989), **"Acoustic Measurement of River Discharge"**, *Journal of Hydraulic Engineering*, Vol 115, no. 7, July 1989, pp 925-936

Münchow, A., Coughran, C. S., Hendershott, M. C., and Winant, C. D. (1995), **"Performance and Calibration of an Acoustic Doppler Current Profiler Towed below the Surface"**, *Journal of Atmospheric and Oceanic Technology*, Vol. 12, April 1995, p435-444

New, A. L. (1992), **"Factors affecting the quality of shipboard ADCP data"**, *Deep Sea Research*, Vol. 39, pp. 1985-1992.

Simpson, M. R. and Oltmann, R. N. (1990), **"An Acoustic Doppler Discharge Measurement System"**, *Proceedings of the 1990 National Conference on Hydraulic Engineering*, Vol 2, pp 903-908

FAP24 (1993) **"Selection of survey techniques"**, *FAP24 report presented to the International Workshop on 'Morphological Behaviour of the Major Rivers in Bangladesh*, Dhaka, November 1993.

APPENDIX A

ADCP ASCII Output Format

Row	Field	Description
1	1	ENSEMBLE TIME – Year (at start of ensemble)
	2	– Month
	3	– Day
	4	– Hour
	5	– Minute
	6	– Second
	7	– Hundredths of seconds
	8	ENSEMBLE NUMBER (or SEGMENT NUMBER for processed or averaged raw data)
	9	NUMBER OF ENSEMBLES IN SEGMENT (if averaging ON or processing data)
	10	PITCH – Average for this ensemble (degrees)
	11	ROLL – Average for this ensemble (degrees)
	12	CORRECTED HEADING – Average ADCP heading + heading offset - magnetic variation
	13	ADCP TEMPERATURE – Average for this ensemble (°C)
2	1	BOTTOM-TRACK VELOCITY – East(+)/West(-); average for this ensemble (cm/s or ft/s)
	2	– North(+)/South(-)
	3	– Vertical (up[+]/down[-])
	4	– Error
	5	WATER LAYER VELOCITY – East(+)/West(-); average for this ensemble (cm/s or ft/s)
	6	– North(+)/South(-)
	7	– Vertical (up[+]/down[-])
	8	– Error
	9	DEPTH READING – Beam 1 average for this ensemble (m or ft, as set by user)
	10	– Beam 2
	11	– Beam 3
	12	– Beam 4
	13	– Beam 5 (if present)
3	1	TOTAL ELAPSED DISTANCE – Through this ensemble (from bottom-track data; in m or ft)
	2	ELAPSED TIME – Through this ensemble (in seconds)
	3	DISTANCED TRAVELLED NORTH (m or ft, as set by user)
	4	DISTANCED TRAVELLED EAST (m or ft, as set by user)
	5	COURSE MADE GOOD – Through this ensemble (from bottom-track data)
4	1	NAVIGATION DATA – Latitude; average for this ensemble (degrees)
	2	– Longitude; average for this ensemble (degrees)
	3	– North(+)/South(-) ship movement; average for this ensemble (cm/s or ft/s)
	4	– East(+)/West(-) ship movement; average for this ensemble (cm/s or ft/s)
	5	– Total distance travelled through this ensemble (cm or ft)
5	1	DISCHARGE VALUES – Middle part of profile (measured); m^3/s or ft^3/s
	2	– Top part of profile (estimated); m^3/s or ft^3/s
	3	– Bottom part of profile (estimated); m^3/s or ft^3/s
	4	– Start-shore discharge estimate; m^3/s or ft^3/s
	5	– Starting distance (boat to shore); m or ft
	6	– End-shore discharge estimate; m^3/s or ft^3/s
	7	– Ending distance (boat to shore); m or ft
	8	– Starting depth of middle layer (or ending depth of top layer); m or ft
	9	– Ending depth of middle layer (or starting depth of bottom layer); m or ft
6	1	NUMBER OF BINS TO FOLLOW
	2	MEASUREMENT UNIT – cm or ft
	3	VELOCITY REFERENCE – BT, LAYER, or NONE
	4	INTENSITY UNITS – dB or counts
	5	INTENSITY SCALE FACTOR – in dB/count
	6	SOUND ABSORPTION FACTOR – in dB/m
7	1	DEPTH – Corresponds to depth of data for present bin (depth cell); includes ADCP depth and blanking value; in m or ft.
	2	VELOCITY MAGNITUDE
	3	VELOCITY DIRECTION
	4	EAST VELOCITY COMPONENT – East(+)/West(-)
	5	NORTH VELOCITY COMPONENT – North(+)/South(-)
	6	VERTICAL VELOCITY COMPONENT – Up(+)/Down(-)
	7	ERROR VELOCITY
	8	ECHO INTENSITY – Beam 1
	9	– Beam 2
	10	– Beam 3
	11	– Beam 4
	12	PERCENT-GOOD
	13	DISCHARGE

APPENDIX B

GTSQ ver 1.0 SOURCE CODE

```
c a title subroutine

      call clearscreen

      write(*, *) '                Welcome to GPSQ ver 1.0 for DOS'
      do k=1, 10
        write (*, *)
      enddo
      write (*, *) 'press ENTER to continue'
      read (*, *)

      call filename

      end

c-----------------------------------------------------------------

      subroutine filename

c a subroutine for entering adcp and navigation input file names

      character*70 adcp_file, nav_file

      call clearscreen

      write(*, *) 'please enter ADCP ascii file name and path'
      write(*, *)
      read(*, '(a70)') adcp_file
      write(*, *)
      call clearscreen
      write(*, *) 'please enter NAVIGATION file name and path'
      write(*, *)
      read(*, '(a70)') nav_file
      call clearscreen

      call data_out(adcp_file, nav_file)

      end

c-----------------------------------------------------------------

      subroutine data_out(adcp_file, nav_file)

c a subroutine for selecting the units of measurement for data output

      real units, units2, units3

      integer option4, error
```

```
      character*70 ft_cm*1, adcp_file, nav_file

      open (4, file = adcp_file, status = 'old')

      call clearscreen

   10 continue
      write(*, *) 'OPTIONS FOR DATA OUTPUT....'
      write(*, *)
      write(*, *) '(1) ENGLISH units (ft, ft/s, cfs)'
      write(*, *)
      write(*, *) '(2) SI units (m, m/s, cumecs)'
      do j=1, 10
       write(*, *)
      enddo
      write(*, *) 'please enter an option number'
      read *, option4

   20 continue

c 'units' is a factor used to convert data into the desired output

      select case(option4)
       case(1)
        units = 3.281
       case(2)
        units = 1.0
       case default
        call clearscreen
        write (*, *) 'please re-enter an option number'
        goto 10
      end select

      do 30, j=1, 8
        read (4, 100)
   30 continue
  100 format(70x)

c Adcp data file is read to determine the measurement unit 'ft_cm'
c 'units2' is used to convert the ADCP data into a form whereby
c 'units' can be applied
c 'units3' is req'd because if the measurement unit is SI then velocity
c data is given in cm not m

      read (4, '(3x, a1)', end=40, err=40, iostat=error) ft_cm
      select case(ft_cm)
       case('f')
        units2 = units/3.281
        units3 = 1
```

```
      case('c')
        units2 = units/1.0
        units3 = 0.01
      end select

      close (4)

      call q_method(units, units2, units3, adcp_file, nav_file)

   40 continue
      if(error .ne. 0)then
        call clearscreen
        write (*,*) 'CANNOT READ measurement unit'
        write (*,*) 'ERROR in', adcp_file
        stop
      endif

      end

c-----------------------------------------------------------------------

c a subroutine for entering preference of power or constant method for
c top layer discharge estimation

      subroutine q_method(units, units2, units3, adcp_file, nav_file)

      real units, units2, units3

      integer option2

      character*70, adcp_file, nav_file

      call clearscreen

   10 continue
      write(*, *) 'OPTIONS FOR DISCHARGE CALCULATION....'
      write(*, *)
      write(*, *) '(1) use "POWER" method for top layer estimation'
      write(*, *)
      write(*, *) '(2) use "CONSTANT" method for top layer estimation'
      do j = 1, 10
        write(*, *)
      enddo
      write(*, *) 'please enter an option number'
      read *, option2

      select case (option2)
        case(1)
          call discharge1(units, units2, units3, adcp_file, nav_file)
```

```
      case(2)
        call discharge2(units, units2, units3, adcp_file, nav_file)
      case default
        call clearscreen
        write (*, *) 'please re-enter option number'
        goto 10
    end select

    end

c-----------------------------------------------------------------------

c a subroutine for calculating discharge using the power method

    subroutine discharge1(units, units2, units3, adcp_file, nav_file)

    real depth_bin(100), vel_mag(100), vel_dirn(100), corr_w,
   +e_vel(100), n_vel(100), z_vel(100), db_avg(100), depth_avg,
   +error_vel(100), total_binq, toplayer_q, btmlayer_q,
   +a_avg, ensemble_q, line_grad, blank_depth, a(100),
   +power_coeff, bin_vel(100), bin_q(100), adcp_range(100),
   +bin_size, line_dirn, total1, total2, total3, total4, bank_q,
   +bin_velavg, start_depth, start_vel, units, units2, units3,
   +corr_e, corr_n, dist

    integer ic, ic2, ic3, num_ens, ens_num, id

    character*70 label1, label2, adcp_file, nav_file

    ic = 0
    ic2 = 1
    ic3 = 0
    total4 = 0


    call velocity(units, units2, units3, num_ens, line_grad, bin_size,
   +blank_depth, adcp_file, nav_file)

    close (1)

    open (1, file = 'velocity.dat', status = 'old')
    open (2, file = 'gpscorrq.out', status = 'unknown')

    read(1, '(a70)') label1
    write(2, *) label1
    read(1, '(a70)') label2
    write(2, *) label2
    read(1, *)
```

```
      call clearscreen
      write(*, *) 'PLEASE ENTER A POWER LAW CO-EFFICIENT'
      write (*, *)
      read *, power_coeff
      call clearscreen

      write (2, *) 'top layer discharge estimation method: power'
      write (2, *) 'power co-efficient =', power_coeff
      write (2, *) 'ensemble, dist, easting, northing, width, depth, bin
     + q, top q, btm q, total q'
      write (2, *) 'power co-efficient =', power_coeff

      depth_bin(0) = blank_depth - bin_size/2

   10 continue
      total1 = 0
      total2 = 0
      total3 = 0

   20 continue
      ic = ic + 1
      read (1, *, err=30, end=30) id, ens_num, dist, corr_e,
     +corr_n, corr_w, depth_avg, depth_bin(ic), vel_mag(ic),
     +vel_dirn(ic), e_vel(ic), n_vel(ic), z_vel(ic), error_vel(ic),
     +db_avg(ic), adcp_range(ic)
      if (id .eq. ic2) then
        goto 20
      endif
      do k=1, 2
        backspace (1)
      enddo
      read (1, *, err=30, end=30) id, ens_num, dist, corr_e, corr_n,
     +corr_w, depth_avg

c 'a( )' is equal to the 'non-adcp' parameters in the power law equation

   30 continue
      ic = ic - 1
      line_dirn =((atan(1/line_grad)) * 180/3.14159) + 90

      do 40, k=1, ic
        bin_vel(k) = vel_mag(k)*cos((line_dirn-vel_dirn(k))*3.14159/180)
        bin_q(k) = bin_vel(k)*corr_w*(depth_bin(k) - depth_bin(k-1))
        a(k) = bin_vel(k)/((depth_avg - depth_bin(k))**power_coeff)
        total1 = bin_q(k) + total1
        total2 = a(k) + total2
        total3 = bin_vel(k) + total3
   40 continue
```

```
      ic3 = ic3 + 1
      total_binq = total1
      a_avg = total2/ic
      bin_velavg = total3/ic
      toplayer_q = corr_w * blank_depth * a_avg *
     +((depth_avg - blank_depth/2)**power_coefl)
      btmlayer_q = a_avg * ((((depth_avg -
     +adcp_range(ic))**7)/2)**power_coefl)*corr_w
      ensemble_q = total_binq + toplayer_q + btmlayer_q


      write (2, 100) ens_num, dist, corr_e, corr_n,
     +corr_w, depth_avg, total_binq, toplayer_q,
     +btmlayer_q, ensemble_q, a_avg
  100 format (i5, f8.2, 2f13.2, 7f10.2)

      if (ic2 .eq. 1) then
        start_depth = depth_avg
        start_vel = total3/ic
      endif

      total4    ensemble_q + total4
      if (ic3 .eq. num_ens) then
        goto 50
      endif
      ic = 0
      ic2 = ic2    1

      goto 10

   50 continue

      end_depth = depth_avg
      end_vel = bin_velavg

      call nearshore(bank_q, start_depth, start_vel, end_depth, end_vel,
     +units)

      call clearscreen
      total4 = bank_q + total4

      if (units .eq. 1.0) then
        write (2, *) 'total discharge = ', total4, '(cumecs)'
        write(*, *) 'TOTAL DISCHARGE = ', total4, '(cumecs)'
        do k = 1, 10
          write(*, *)
        enddo
      end if
```

```
      if (units .eq. 3.281) then
        write (2, *) 'total discharge = ', total4, '(cfs)'
        write(*, *) 'TOTAL DISCHARGE = ', total4, '(cfs)'
        do k=1, 10
          write(*, *)
        enddo
      end if

   60 continue

      end

c-----------------------------------------------------------------------

c a subroutine for calculating discharge using the constant method
c a power law fit is still used for the near-bed discharge, as such
c 'discharge2' is very similar to 'discharge1'

      subroutine discharge2(units, units2, units3, adcp_file, nav_file)

      real depth_bin(100), vel_mag(100), vel_dirn(100), corr_w,
     +e_vel(100), n_vel(100), z_vel(100), db_avg(100), depth_avg,
     +error_vel(100), total_binq, toplayer_q, btmlayer_q,
     +a_avg, ensemble_q, line_grad, blank_depth, a(100),
     +power_coeff, bin_vel(100), bin_q(100), adcp_range(100),
     +bin_size, line_dirn, total1, total2, total3, total4, bank_q,
     +bin_velavg, vel_top, start_depth, start_vel, units, units2,
     +units3, corr_e, corr_n, dist

      integer ic, ic2, ic3, num_ens, ens_num, id

      character*70 label1, label2, adcp_file, nav_file

      ic = 0
      ic2 = 1
      ic3 = 0
      total4 = 0

      call velocity(units, units2, units3, num_ens, line_grad, bin_size,
     +blank_depth, adcp_file, nav_file)

      close (1)

      open (1, file = 'velocity.dat', status = 'old')
      open (2, file = 'gpscorrq.out', status = 'unknown')

      read(1, '(a70)') label1
      write(2, *) label1
      read(1, '(a70)') label2
```

```
  write(2, *) label2
  read(1, *)

  depth_bin(0) = blank_depth - bin_size/2

  call clearscreen
  write(*, *) 'Please enter a power law co-efficient '
  write (*, *)
  read *, power_coeff
  call clearscreen

  write (2, *) 'top layer discharge estimation method: constant'
  write (2, *) 'power coefficient for bottom layer =', power_coeff
  write (2, *) 'ensemble, dist, easting, northing, width, depth, bin
 + q, top q, btm q, total q'


10 continue
  total1 = 0
  total2 = 0
  total3 = 0

20 continue
  ic = ic + 1
  read (1, *, err=30, end=30) id, ens_num, dist, corr_e, corr_n,
 +corr_w, depth_avg, depth_bin(ic), vel_mag(ic), vel_dirn(ic),
 +e_vel(ic), n_vel(ic), z_vel(ic), error_vel(ic), db_avg(ic),
 +adcp_range(ic)
  if (id .eq. ic2) then
   goto 20
  endif
  do k=1, 2
   backspace (1)
  enddo
  read (1, *, err=30, end=30) id, ens_num, dist, corr_e, corr_n,
  +corr_w, depth_avg

30 continue
  ic = ic - 1
  line_dirn = ((atan(1/line_grad))*180/3.14159) + 90
  do 40, k=1, ic
   bin_vel(k) = vel_mag(k)*cos((line_dirn-vel_dirn(k))*3.14159/180)
   bin_q(k) = bin_vel(k) * corr_w * (depth_bin(k)-depth_bin(k-1))
   a(k) = bin_vel(k)/((depth_avg - depth_bin(k))**power_coeff)
   total1 = bin_q(k) + total1
   total2 = a(k) + total2
   total3 = bin_vel(k) + total3
40 continue
```

```
      vel_top = vel_mag(1)*cos((line_dirn - vel_dirn(1))*3.14159/180)
      ic3 = ic3 + 1
      total_binq = total1
      a_avg = total2/ic
      bin_velavg = total3/ic

      toplayer_q = corr_w * blank_depth * vel_top

      btmlayer_q = a_avg * ((((depth_avg -
     +adcp_range(ic))**7)/2)**power_coeff) * corr_w

      ensemble_q = total_binq + toplayer_q + btmlayer_q

      write (2, 100) ens_num, dist, corr_e, corr_n, corr_w, depth_avg,
     +total_binq, toplayer_q, btmlayer_q, ensemble_q, a_avg
  100 format (i5, f8.2, 2f13.2, 7f10.2)

      if (ic2 .eq. 1) then
        start_depth = depth_avg
        start_vel = total3/ic
      endif

      total4 =  ensemble_q + total4
      if (ic3 .eq. num_ens) then
        goto 50
      endif
      ic = 0
      ic2 = ic2 + 1

      goto 10

   50 continue

      end_depth = depth_avg
      end_vel = bin_velavg

      call nearshore(bank_q, start_depth, start_vel, end_depth, end_vel,
     +units)

      call clearscreen
      total4 = bank_q + total4

      if (units .eq. 1.0) then
        write (2, *) 'total discharge = ', total4, '(cumecs)'
        write(*, *) 'TOTAL DISCHARGE = ', total4, '(cumecs)'
        do k=1, 10
          write(*, *)
        enddo
      endif
```

```
      if (units .eq. 3.281) then
        write (2, *) 'total discharge = ', total4, '(cfs)'
        write(*, *) 'TOTAL DISCHARGE = ', total4, '(cfs)'
        do k=1, 10
          write(*, *)
        enddo
      end if

   60 continue

      end

c-----------------------------------------------------------------------
c a subroutine used to calculate the missing discharge, if any, at the
c start and end of a transect

      subroutine nearshore(bank_q, start_depth, start_vel, end_depth,
     +end_vel, units)

      real bank_q, start_dist, end_dist, start_q, end_q, start_depth,
     +start_vel, end_depth, end_vel, units

      integer option3

      call clearscreen

   05 continue
      write(*, *) 'NEAR SHORE DISCHARGE OPTIONS...'
      write(*, *)
      write(*, *) '(1) CALCULATE near shore discharges'
      write(*, *)
      write(*, *) '(2) DO NOT CALCULATE near shore discharges'
      write(*, *)
      write(*, *) 'please enter an option number'
      read *, option3

      select case(option3)
       case(1)
         goto 10
       case(2)
         goto 20
       case default
         call clearscreen
         write (*, *) 'please re-enter option number'
         goto 05
      end select


   10 continue
```

```
      call clearscreen

      if (units .eq. 1.0) then
       write(*, *) 'enter distance to bank at START of transect (m)'
       read *, start_dist
       write(*, *)
       write(*, *) 'enter distance to bank at END of transect (m)'
       read *, end_dist
      end if

      if (units .eq. 3.281) then
       write(*, *) 'enter distance to bank at START of transect (ft)'
       read *, start_dist
       write(*, *)
       write(*, *) 'enter distance to bank at END of transect (ft)'
       read *, end_dist
      end if

      call clearscreen

      start_q = 0.707 * start_depth * start_dist * start_vel/2

      end_q = 0.707 * end_depth * end_dist * end_vel/2

      bank_q = start_q + end_q

      goto 30

   20 continue
      bank_q = 0

   30 continue

      end

c-----------------------------------------------------------------------

      subroutine velocity(units, units2, units3, num_ens, line_grad,
     +bin_size, blank_depth, adcp_file, nav_file)

c a  subroutine for combining the vessel velocity and the adcp measured
c water velocities

      real depth(100), mag_adcp(100), dirn_adcp(100), e_adcp(100),
     +n_adcp(100), z_adcp(100), error_vel(100), db_1(100), db_2(100),
     +db_3(100), db_4(100), per_good(100), e_boat, n_boat, corr_e,
     +corr_n, corr_w, depth_avg, dist, e_water(100), n_water(100),
     +z_water(100), mag_water(100), dirn_water(100), db_avg(100),
     +junk1(100), junk2(100), junk3(100), line_grad, bin_size,
```

```
     +adcp_angle, adcp_range, bin_ratio, blank_depth, mag_decln,
     +blank_dist, units, units2, units3, junk4

      integer ens_num, good_bins, numbins, ensemblenum, num_ens, ic,
     +ic2, error, error2, error3, option1

      character*70 label1, label2, adcp_file, nav_file

      ic = 0
      ic2 = 0

      call clearscreen
    05 continue
      write(*, *) 'DEPTH CALCULATION OPTIONS... '
      write(*, *)
      write(*, *) '(1) use ADCP beam depths'
      write(*, *)
      write(*, *) '(2) use FATHOMETER data'
      do j=1, 10
        write(*, *)
      enddo
      write(*, *) 'please enter an option number'
      read *, option1

      select case (option1)
        case (1)
          call ens_header1(units, num_ens, line_grad, nav_file)
        case (2)
          call ens_header2(units, num_ens, line_grad, nav_file)
        case default
          call clearscreen
          write (*, *) 'please re-enter option number'
          goto 05
      end select

      rewind (3)

      open (1, file = 'velocity.dat', status = 'unknown')
      open (4, file = adcp_file, status = 'old')

      call clearscreen

      write(*, *) 'please enter ADCP BEAM ANGLE (deg)'
      write (*, *)
      read *, adcp_angle
      call clearscreen
      write(*, *) 'if NOT already corrected...'
      write(*, *) 'please enter the local magnetic declination (deg)'
      write (*, *)
```

```
      read *, mag_decln

      call clearscreen

      read(4, '(a70)') label1
      read(4, '(a70)') label2
      write(1, *) label1
      write(1, *) label2

      write(1, 100) 'id, ens, dist, east, north, width, depth, bin, vel
     +mag, vel dirn, east vel, north vel, vert vel, error vel, db_avg, a
     +dcp_range'
  100 format(a120)

      read(4, *, err=80, end=95, iostat=error) bin_size, junk4,
     +adcp_depth
      bin_size = (bin_size/100) * units
      adcp_depth = (adcp_depth/100) * units

   10 continue
      ic = 0

   20 continue
      ic2 = ic2 + 1
      read (4, *, err=80, end=95, iostat=error) (junk1(i), i=1, 7),
     +ensemblenum, (junk2(i), i=1, 13), depth_b1, depth_b2, depth_b3,
     +depth_b4, (junk3(i), i=1, 19), numbins
      depth_avg = (depth_b1 + depth_b2 + depth_b3 + depth_b4)/4
      depth_avg = depth_avg * units2

   30 continue
      if (option1 .eq. 1) then
        read (3, *, err=85, end=95, iostat=error2) ens_num, dist,
     +  corr_e, corr_n, corr_w, e_boat, n_boat
      endif

      if (option1 .eq. 2) then
        read (3, *, err=85, end=95, iostat=error2) ens_num, dist,
     +  corr_e, corr_n, corr_w, e_boat, n_boat, depth_avg
      endif

      if (ens_num .lt. ensemblenum) then
        goto 30
      endif

   40 continue
      ic = ic + 1
      read (4, *, end=50, err=90, iostat=error3) depth(ic),
     +mag_adcp(ic), dirn_adcp(ic), e_adcp(ic), n_adcp(ic), z_adcp(ic),
```

```
     +error_vel(ic), db_1(ic), db_2(ic), db_3(ic), db_4(ic),
     +per_good(ic)
      if (ic .ge. numbins) goto 50
      goto 40

   50 continue
      if (ens_num .gt. ensemblenum) then
       goto 20
      endif

      do j=1, ic
       depth(j) = depth(j) * units2
       mag_adcp(j) = mag_adcp(j) * units2 * units3
       dirn_adcp(j) = dirn_adcp(j) - mag_deeln
       if(dirn_adcp(j) .gt. 360) then
         dirn_adcp(j) = dirn_adcp(j) - 360
       endif
       z_adcp(j) = z_adcp(j) * units2 * units3
       error_vel(j) = error_vel(j) * units2 * units3
      enddo

      do 60, k=1, ic
       e_adcp(k) = mag_adcp(k) * sin((dirn_adcp(k))*3.14159/180)
       n_adcp(k) = mag_adcp(k) * cos((dirn_adcp(k))*3.14159/180)
       e_water(k) = e_boat + e_adcp(k)
       n_water(k) = n_boat + n_adcp(k)
       z_water(k) = z_adcp(k)
       mag_water(k) =(e_water(k)**2 + n_water(k)**2)**0.5
       dirn_water(k) = (acos(n_water(k)/mag_water(k)))*180/3.14159
       db_avg(k) = (db_1(k)+db_2(k)+db_3(k)+db_4(k))/4
   60 continue


      adcp_range = (depth_avg - adcp_depth)*(cos((adcp_angle)*
     +3.14159/180)) + adcp_depth
      blank_dist = depth(1) - bin_size/2 - adcp_depth
      blank_depth = adcp_depth + blank_dist
      bin_range = adcp_range - blank_depth
      bin_ratio = bin_range/bin_size
      good_bins = aint(bin_ratio)

      do 70, k = 1, good_bins
       if (mag_water(k) .gt. 95) then
         goto 70
       endif
       write (1, 200) ic2, ens_num, dist, corr_e, corr_n, corr_w,
     + depth_avg, depth(k), mag_water(k), dirn_water(k), e_water(k),
     + n_water(k), z_water(k), error_vel(k), db_avg(k), adcp_range
   70 continue
```

```
200 format (2i5, f8.2, 2f13.2, 11f10.2)
   goto 10

 80 continue
   if (error .ne. 0) then
    write(*, *) 'error in adcp header, ensemble no.', ic2
    stop
   endif

 85 continue
   if (error2 .ne. 0) then
    write(*, *) 'error in navfile, ensemble no.', ic2
    stop
   endif

 90 continue
   if (error3 .gt. 0) then
    write(*, *) 'error in adcp bin data, ensemble no.', ic2
    stop
   endif

 95 continue

   end

c-----------------------------------------------------------------------

    subroutine ens_header1(units, num_ens, line_grad, nav_file)

c a subroutine for calculating the survey line definintion, corrected
c ensemble positions and widths, and vessel velocities

    real e_end(100), n_end(100), e_vel, n_vel, e_start, n_start,
   +line_grad,  line_int, corr_e, corr_n, corr_w, sum_e,
   +sum_esq, sum_n, sum_en, corr_estart, corr_nstart, last_e, last_n,
   +corr_eavg, corr_navg, units, dist

    integer ic, ens_num(100), num_ens, time(100), time_start

    character*70 nav_file

    sum_e = 0
    sum_esq = 0
    sum_n = 0
    sum_en = 0
    last_e = 0
    last_n = 0
    ic = 0
```

```
      call read_gps1(num_ens, time_start, e_start, n_start, time,
     +ens_num, e_end, n_end, nav_file)

      open (3, status = 'scratch')

      sum_e = e_start + sum_e
      sum_esq = e_start**2 + sum_esq
      sum_n = n_start + sum_n
      sum_en = e_start*n_start + sum_en

      ic = num_ens + ic
      do 10 j=1, ic
        sum_e = sum_e + e_end(j)
        sum_esq = sum_esq + e_end(j)**2
        sum_n = sum_n + n_end(j)
        sum_en = sum_en + e_end(j)*n_end(j)
   10 continue

      ic = ic + 1
      line_grad = (ic*sum_en - sum_e*sum_n)/(ic*sum_esq - sum_e**2)
      line_int = (sum_esq*sum_n - sum_e*sum_en)/(ic*sum_esq - sum_e**2)
      ic = ic - 1

      e_vel = (e_end(1) - e_start)*units/(time(1) - time_start)
      n_vel = (n_end(1) - n_start)*units/(time(1) - time_start)

      corr_estart = (line_grad*n_start + e_start - line_grad*line_int)
     +/(line_grad**2 + 1)
      corr_nstart = line_grad*corr_estart + line_int

      corr_e = (line_grad*n_end(1) + e_end(1) - line_grad*line_int)
     +/(line_grad**2 + 1)
      corr_n = line_grad*corr_e + line_int

      corr_w = (((corr_e - corr_estart)**2 + (corr_n - corr_nstart)**2)
     +**0.5)*units

      dist = corr_w/2

      corr_eavg = (corr_estart + corr_e)/2
      corr_navg = (corr_nstart + corr_n)/2

      write(3, *) ens_num(1), dist, corr_eavg, corr_navg, corr_w, e_vel,
     +n_vel

      last_e = corr_e
      last_n = corr_n

      do 20, j=2, ic
```

```
      e_vel = (e_end(j) - e_end(j-1))*units/(time(j) - time(j-1))
      n_vel = (n_end(J) - n_end(J-1))*units/(time(J) - time(j-1))

      corr_e = (line_grad*n_end(j) + e_end(j) - line_grad*line_int)
     + /(line_grad**2 + 1)
      corr_n = line_grad*corr_e + line_int

      corr_w = (((corr_e - last_e)**2 + (corr_n - last_n)**2)**0.5)*
     + units

      dist = (((corr_e - corr_estart)**2 + (corr_n - corr_nstart)**2)
     + **0.5)*units - corr_w/2

      corr_eavg = (corr_e + last_e)/2
      corr_navg = (corr_n + last_n)/2

      write (3, *) ens_num(j), dist, corr_eavg, corr_navg, corr_w,
     + e_vel, n_vel

      last_e = corr_e
      last_n = corr_n
   20 continue

      end

c-----------------------------------------------------------------------

      subroutine ens_header2(units, num_ens, line_grad, nav_file)

c a subroutine for calculating the survey line definintion, corrected
c ensemble positions, widths, depths and vessel velocities


      real e_end(100), n_end(100), e_vel, n_vel, e_start, n_start,
     +line_grad, line_int, corr_e, corr_n, corr_w, sum_e,
     +sum_esq, sum_n, sum_en, corr_estart, corr_nstart, last_e, last_n,
     +corr_eavg, corr_navg, units, dist, depth_avg(100)

      integer ic, ens_num(100), num_ens, time(100), time_start

      character*70 nav_file

      sum_e = 0
      sum_esq = 0
      sum_n = 0
      sum_en = 0
      last_e = 0
      last_n = 0
      ic = 0
```

```
      call read_gps2(num_ens, time_start, e_start, n_start, time,
     +ens_num, e_end, n_end, depth_avg, nav_file)

      open (3, status = 'scratch')

      sum_e = e_start + sum_e
      sum_esq = e_start**2 + sum_esq
      sum_n = n_start + sum_n
      sum_en = e_start*n_start + sum_en

      ic = num_ens + ic
      do 10 j=1, ic
        sum_e = sum_e + e_end(j)
        sum_esq = sum_esq + e_end(j)**2
        sum_n = sum_n + n_end(j)
        sum_en = sum_en + e_end(j)*n_end(j)
   10 continue

      ic = ic + 1
      line_grad = (ic*sum_en - sum_e*sum_n)/(ic*sum_esq - sum_e**2)
      line_int = (sum_esq*sum_n - sum_e*sum_en)/(ic*sum_esq - sum_e**2)
      ic = ic - 1

      do 15, j = 1, ic
        depth_avg(1) = depth_avg(j) * units
   15 continue


      e_vel = (e_end(1) - e_start)*units/(time(1) - time_start)
      n_vel = (n_end(1) - n_start)*units/(time(1) - time_start)

      corr_estart = (line_grad*n_start + e_start - line_grad*line_int)
     +/(line_grad**2 + 1)
      corr_nstart = line_grad*corr_estart + line_int

      corr_e = (line_grad*n_end(1) + e_end(1) - line_grad*line_int)
     +/(line_grad**2 + 1)
      corr_n = line_grad*corr_e + line_int

      corr_w = (((corr_e - corr_estart)**2 + (corr_n - corr_nstart)**2)
     +**0.5)*units

      dist = corr_w/2

      corr_eavg = (corr_estart + corr_e)/2
      corr_navg = (corr_nstart + corr_n)/2

      write(3, *) ens_num(1), dist, corr_eavg, corr_navg, corr_w, e_vel,
```

```
     +n_vel, depth_avg(1)

      last_e = corr_e
      last_n = corr_n

      do 20, j=2, ic
       e_vel = (e_end(j) - e_end(j-1))*units/(time(j) - time(j-1))
       n_vel = (n_end(J) - n_end(J-1))*units/(time(J) - time(j-1))

       corr_e = (line_grad*n_end(j) + e_end(j) - line_grad*line_int)
     + /(line_grad**2 + 1)
       corr_n = line_grad*corr_e + line_int

       corr_w = (((corr_e - last_e)**2 + (corr_n - last_n)**2)**0.5)*
     + units

       dist = (((corr_e - corr_estart)**2 + (corr_n - corr_nstart)**2)
     + **0.5)*units - corr_w/2

       corr_eavg = (corr_e + last_e)/2
       corr_navg = (corr_n + last_n)/2

       write (3, *) ens_num(j), dist, corr_eavg, corr_navg, corr_w,
     + e_vel, n_vel, depth_avg(j)

       last_e = corr_e
       last_n = corr_n
     20 continue

       end


c-----------------------------------------------------------------------

       subroutine read_gps1(ic4, time_start, e_start, n_start,
      +ens_time, ens_num, e_end, n_end, nav_file)

c a subroutine for extracting ensemble data from gps file which
c does not include depth data

       real e_start, n_start, e_end(100), n_end(100), gps_e(1000),
      +gps_n(1000)

       integer time_start, hour_start, min_start, sec_start,
      +ens_time(100), gps_hour(1000), gps_min(1000), gps_sec(1000), ic,
      +ic4, error, ens_num(100)

       character*1 chk, nav_file*70
```

```
  ic = 0
  ic4 = 0

  open (5, file = nav_file, status ='old')

01 continue
  read (5, '(a1)') chk
  select case(chk)
   case('E')
    goto 02
   case('e')
    goto 02
   case default
    goto 01
  end select

02 continue
  read (5, 200, err=20, end=20, iostat=error) hour_start, min_start,
 +sec_start, e_start, n_start

  time_start = hour_start*3600 + min_start*60 + sec_start

  do j=1, 2
   backspace 5
  enddo

05 continue
  ic4 = ic4 + 1
  read (5, 100, err=40, end=40) ens_num(ic4)
10 continue
  ic = ic + 1
  read (5, 200, err=30, end=30, iostat=error) gps_hour(ic),
 +gps_min(ic), gps_sec(ic), gps_e(ic), gps_n(ic)

  goto 10

100 format(10x, i4)
200 format(i2, 1x, i2, 1x, i2, f12.2, f10.2)

 20 continue
  if (error .ne. 0) then
   write(*, *) 'error in gps data'
   stop
  endif

 30 continue
  if (error .ne. 0) then
   ic = ic - 2
   ens_time(ic4) = gps_hour(ic)*3600 + gps_min(ic)*60 + gps_sec(ic)
```

```
      if(ens_time(ic4).eq.0) then
       e_end(ic4) = e_end(ic4-1) + (e_end(ic4-1) - e_end(ic4-2))
       n_end(ic4) = n_end(ic4-1) + (n_end(ic4-1) - n_end(ic4-2))
       ens_time(ic4) = 2*ens_time(ic4-1) - ens_time(ic4-2)
       write(*, *) 'error at ens_num', ens_num(ic4)
       read(*, *)
       ic4 = ic4 + 1
       goto 40
      endif
      e_end(ic4) = gps_e(ic)
      n_end(ic4) = gps_n(ic)
      backspace 5
      goto 05
     endif

  40 continue

     ic4 = ic4 - 1

     end

c-----------------------------------------------------------------------

     subroutine read_gps2(ic4, time_start, e_start, n_start,
    +ens_time, ens_num, e_end, n_end, depth_avg, nav_file)

c a subroutine for extracting ensemble data from gps file which includes
c depth data

     real e_start, n_start, e_end(100), n_end(100), gps_e(1000),
    +gps_n(1000), depth_total, depth_avg(100), depth(1000)

     integer time_start, hour_start, min_start, sec_start,
    +ens_time(100), gps_hour(1000), gps_min(1000), gps_sec(1000), ic,
    +ic2, ic3, ic4, error, ens_num(100)

     character*1 chk, nav_file*70

     ic = 0
     ic2 = 0
     ic3 = 0
     ic4 = 0

     open (5, file = nav_file, status = 'old')

  01 continue
     read (5, '(a1)') chk
     select case(chk)
      case('E')
```

```
      goto 02
    case('e')
      goto 02
    case default
      goto 01
  end select

02 continue
  ic2 = ic2 + 1
  read (5, 200, err=20, end=20, iostat=error) hour_start, min_start,
 +sec_start, n_start, e_start

  time_start = hour_start*3600 + min_start*60 + sec_start

  do j=1, 2
   backspace 5
  enddo

05 continue
  ic4 = ic4 + 1
  read (5, 100, err=40, end=40) ens_num(ic4)
10 continue
  ic = ic + 1
  read (5, 200, err=30, end=30, iostat=error) gps_hour(ic),
 +gps_min(ic), gps_sec(ic), gps_n(ic), gps_e(ic), depth(ic)

  goto 10

100 format(10x, i4)
200 format(i2, 1x, i2, 1x, i2, f12.2, f10.2, f10.2)

 20 continue
  if (error .ne. 0) then
   write(*, *) 'error in gps data'
   stop
  endif

 30 continue
  if (error .ne. 0) then
   ic = ic - 2
   depth_total = 0

   do j=ic2, ic
    depth_total = depth(j) + depth_total
    ic3 = ic3 + 1
   enddo

   depth_avg(ic4) = depth_total/ic3
   ens_time(ic4) = gps_hour(ic)*3600 + gps_min(ic)*60 + gps_sec(ic)
```

```
      e_end(ic4) = gps_e(ic)
      n_end(ic4) = gps_n(ic)

      ic2 = ic + 1
      ic3 = 0
      backspace 5
      goto 05
     endif

   40 continue

     ic4 = ic4 - 1

     end

c------------------------------ ------------------------------------

     subroutine clearscreen

     do k=1, 25
      write(*, *)
     enddo

     end

c-----------------------------------------------------------------------
c                    }---<<..THE.END..>>---{
c-----------------------------------------------------------------------
```

APPENDIX C

ADCPFILL SOURCE CODE

```
c a subroutine for filling in bad ensemble data

      real bdepth1(10), bdepth2(10), bdepth3(10), headа_1(20),
     +headа_2(20), headа_3(20), headb_1(30), headb_2(30), headb_3(30),
     +topbtm1(10), topbtm2(10), topbtm3(10)

      real depth1(100), depth2(100), depth3(100), vel_mag1(100),
     +vel_mag2(100), vel_mag3(100), vel_dirn1(100), vel_dirn2(100),
     +vel_dirn3(100), east_vel1(100), east_vel2(100), east_vel3(100),
     +north_vel1(100), north_vel2(100), north_vel3(100), vert_vel1(100),
     +vert_vel2(100), vert_vel3(100), error_vel1(100), error_vel2(100),
     +error_vel3(100), db_11(100), db_12(100), db_13(100), db_21(100),
     +db_22(100), last_binq, q_diff,
     +db_23(100), db_31(100), db_32(100), db_33(100), db_41(100),
     +db_42(100), db_43(100), discharge1(100), discharge2(100),
     +discharge3(100), last_t, dt, bin_size, units, e_total,
     +n_total, bin_total, l_total, de, dn, dl, dbin_q, dtop_q, dbtm_q


      integer time1(10), time2(10), time3(10), ens_num1, ens_num2,
     +ens_num3, ic, ic2, ic3, ic4, num_bins, per_good1(100),
     +per_good2(100), per_good3(100)

      character*70 label1, label2, label3, label4*59, file_name, ft_em*1

      ic = 0
      ic2 = 0
      last_binq = 1
      last_top = 0
      last_btm = 0
      de = 0
      dn = 0
      dl = 0
      dbin_q = 0
      dtop_q = 0
      dbtm_q = 0

      call clearscreen
      write(*, *) '              Welcome to ADCPFILL for DOS'
      do k=1, 10
        write(*, *)
      enddo
      write(*, *) 'press ENTER to continue'
      read(*,*)

   05 continue
      call clearscreen
      write(*, *) 'please enter adcp ascii file name and path'
      read(*, '(a50)', end=05, err=05) file_name
```

```
      call clearscreen
      write(*, *) 'please wait while ADCPFILT checks...'
      write(*, *) '                ', file_name
      do k=1, 10
       write(*, *)
      enddo
      open(1, file = file_name, status = 'old')
      open(2, file = 'clean.out', status = 'unknown')
      open(3, status = 'scratch')

      read(1, '(a70)', end=90, err=90) label1
      write(2, *) label1
      read(1, '(a70)', end=90, err=90) label2
      write(2, *) label2
      read(1, *) bin_size
      backspace 1
      read(1, '(a70)', end=90, err=90) label3
      write(2, *) label3


   10 continue
      read(1, *, end=90, err=90) (time1(i), i=1, 7),
     +ens_num1, (heada_1(i), i=1, 13), (bdepth1(i), i=1, 4),
     +(headb_1(i), i=1, 19), num_bins


      backspace 1
      read(1, 100, end=90, err=90) ft_cm
  100 format(3x, a1)
      select case(ft_cm)
        case('f')
        units = 3.281
        case('c')
        units = 1.0
      end select
      backspace 1
      read(1, '(a50)', end=90, err=90) label4


      if (bdepth1(1) .lt. 1.0) then
        bdepth1(1) = (bdepth1(2) + bdepth1(3) + bdepth1(4))/3
      endif
      if (bdepth1(2) .lt. 1.0) then
        bdepth1(2) = (bdepth1(1) + bdepth1(3) + bdepth1(4))/3
      endif
      if (bdepth1(3) .lt. 1.0) then
        bdepth1(3) = (bdepth1(1) + bdepth1(2) + bdepth1(4))/3
      endif
      if (bdepth1(4) .lt. 1.0) then
```

```
    bdepth1(4) = (bdepth1(1) + bdepth1(2) + bdepth1(3))/3
   endif

   last_t = headb_1(2)

 20 continue
   ic = ic + 1
   read(1, *, end=90, err=90) depth1(ic), vel_mag1(ic),
  +vel_dirn1(ic), east_vel1(ic), north_vel1(ic), vert_vel1(ic),
  +error_vel1(ic), db_11(ic), db_21(ic), db_31(ic), db_41(ic),
  +per_good1(ic), discharge1(ic)
   if(ic .ge. num_bins) goto 30
   goto 20

 30 continue
   q_diff = headb_1(11) - last_binq
   if(q_diff .eq. 0) then
    ic = 0
    goto 10
   endif

   ic = ic - 1

 35 continue
   write(2, 200) (time1(i), i=1, 7), ens_num1, (heada_1(i), i=1, 13),
  +(bdepth1(i), i=1, 4), (headb_1(i), i=1, 19), label4
   do k=1, num_bins
    write(2, 300)depth1(k), vel_mag1(k), vel_dirn1(k), east_vel1(k),
  +  north_vel1(k), vert_vel1(k), error_vel1(k), db_11(k), db_21(k),
  +  db_31(k), db_41(k), per_good1(k), discharge1(k)
   enddo

 200 format(7i4, i6, 5f8.2, /, 8f10.2, 4f7.2, /, 5f10.2, /, 5f10.2, /,
  + 3f14.2, 6f8.2, /, a50)
 300 format(f9.1, f11.2, f11.3, 8f9.1, i7, f13.2)
   rewind (3)

  40 continue
   last_binq = headb_1(11)
   topbtm1(1) = headb_1(12) - last_top
   topbtm1(2) = headb_1(13) - last_btm
   last_top = headb_1(12)
   last_btm = headb_1(13)
   read(1, *, end=90, err=90) (time2(i), i=1, 7),
  +ens_num2, (heada_2(i), i=1, 13), (bdepth2(i), i=1, 4),
  +(headb_2(i), i=1, 19), num_bins

   if (bdepth2(1) .lt. 1.0) then
    bdepth2(1) = (bdepth2(2) + bdepth2(3) + bdepth2(4))/3
```

```
      endif
      if (bdepth2(2) .lt. 1.0) then
        bdepth2(2)   (bdepth2(1) + bdepth2(3) + bdepth2(4))/3
      endif
      if (bdepth2(3) .lt. 1.0) then
        bdepth2(3)   (bdepth2(1) + bdepth2(2) + bdepth2(4))/3
      endif
      if (bdepth2(4) .lt. 1.0) then
        bdepth2(4)   (bdepth2(1) + bdepth2(2) + bdepth2(3))/3
      endif

      q_diff   headb_2(11) - last_binq + dbin_q
      if(q_diff .ne. 0) then
        goto 60
      endif

      ic2   ic2 + 1

      dt   headb_2(2) - last_t
      last_t   headb_2(2)

      write(3, *) (time2(i), i  1, 7), headb_2(2), dt

      do 50, k  1, num_bins
        read(1, 400, end  90, err  90)
   50 continue
  400 format(70x)

      goto 40

   60 continue

      if(ic2 .gt. 0) goto 70
      time1   time2
      ens_num1   ens_num2
      heada_1   heada_2
      bdepth1   bdepth2
      headb_1   headb_2
      last_t   headb_1(2)

      headb_1(1)   headb_1(1) + dl
      headb_1(3)   headb_1(3) + dn
      headb_1(4)   headb_1(4) + de
      headb_1(5)   (headb_1(3)**2 + headb_1(4)**2)**0.5
      headb_1(11)   headb_1(11) + dbin_q
      headb_1(12)   headb_1(12) + dtop_q
      headb_1(13)   headb_1(13) + dbtm_q

      ic   0
```

```
      goto 20

 70 continue
      call clearscreen
      ic = 0
      bin_total = headb_1(11)
      n_total = headb_1(3)
      e_total = headb_1(4)
      l_total = headb_1(1)
      topbtm2(1) = headb_2(12) - headb_1(12)
      topbtm2(2) = headb_2(13) - headb_1(13)
      rewind (3)

 80 continue
      ic = ic + 1
      read(1, *, end=90, err=90) depth2(ic), vel_mag2(ic),
     +vel_dirn2(ic), east_vel2(ic), north_vel2(ic), vert_vel2(ic),
     +error_vel2(ic), db_12(ic), db_22(ic), db_32(ic), db_42(ic),
     +per_good2(ic), discharge2(ic)
      if(ic .lt. num_bins) then
       goto 80
      endif

      ic3 = ic2
      ic4 = 1
      do 85 k=1, ic2
      topbtm3 = (topbtm1*ic3 + topbtm2*ic4)/(ic3+ic4)
      ens_num3 = (ens_num1*ic3 + ens_num2*ic4)/(ic3+ic4)
      heada_3 = (heada_1*ic3 + heada_2*ic4)/(ic3+ic4)
      bdepth3 = (bdepth1*ic3 + bdepth2*ic4)/(ic3+ic4)
      headb_3 = (headb_1*ic3 + headb_2*ic4)/(ic3+ic4)
      depth3 = depth1
      vel_mag3 = (vel_mag1*ic3 + vel_mag2*ic4)/(ic3+ic4)
      vel_dirn3 = (vel_dirn1*ic3 + vel_dirn2*ic4)/(ic3+ic4)
      east_vel3 = (east_vel1*ic3 + east_vel2*ic4)/(ic3+ic4)
      north_vel3 = (north_vel1*ic3 + north_vel2*ic4)/(ic3+ic4)
      vert_vel3 = (vert_vel1*ic3 + vert_vel2*ic4)/(ic3+ic4)
      error_vel3 = (error_vel1*ic3 + error_vel2*ic4)/(ic3+ic4)
      db_13 = (db_11*ic3 + db_12*ic4)/(ic3+ic4)
      db_23 = (db_21*ic3 + db_22*ic4)/(ic3+ic4)
      db_33 = (db_31*ic3 + db_32*ic4)/(ic3+ic4)
      db_43 = (db_41*ic3 + db_42*ic4)/(ic3+ic4)
      per_good3 = (per_good1*ic3 + per_good2*ic4)/(ic3+ic4)

      read(3, *) (time3(ik), ik=1, 7), headb_3(2), dt
      bin_q = 0
      do i 1, num_bins
       discharge3(i) = ((heada_3(6)*north_vel3(i))-(heada_3(7)*
     + east_vel3(i)))*(bin_size/100)*units*dt
```

```
 if(abs(discharg 3(i)) .lt. 10000) then
  bin_q = discharge3(i) + bin_q
 endif
enddo
headb_3(11) = bin_q + bin_total
headb_3(3) = heada_3(7) * dt + n_total
headb_3(4) = heada_3(6) * dt + e_total
headb_3(5) = (headb_3(3)**2 + headb_3(4)**2)**0.5
headb_3(1) = ((heada_3(7)*dt)**2 + (heada_3(6)*dt)**2)
+**0.5 + l_total
headb_3(12) = topbtm3(1) + last_top
headb_3(13) = topbtm3(2) + last_btm
write(2, 200) (time3(i), i=1, 7), ens_num3, (heada_3(i), i=1, 13),
+(bdepth3(i), i=1, 4), (headb_3(i), i=1, 19), label4
 do j=1, num_bins
 write(2, 300) depth3(j), vel_mag3(j), vel_dirr3(j), east_vel3(j),
+ north_vel3(j), vert_vel3(j), error_vel3(j), db_13(j), db_23(j),
+ db_33(j), db_43(j), per_good3(j), discharge3(j)
 enddo
ic3 = ic3 - 1
ic4 = ic4 + 1
print *, 'ensemble no. ',ens_num3,' has been fixed'
l_total = headb_3(1)
n_total = headb_3(3)
e_total = headb_3(4)
bin_total = headb_3(11)
last_top = headb_3(12)
last_btm = headb_3(13)
85 continue

dl = l_total - headb_1(1)
dn = n_total - headb_1(3)
de = e_total - headb_1(4)
dbin_q = bin_total - headb_1(11)
dtop_q = headb_3(12) - headb_1(12)
dbtm_q = headb_3(13) - headb_1(13)

headb_2(1) = headb_2(1) + dl
headb_2(3) = headb_2(3) + dn
headb_2(4) = headb_2(4) + de
headb_2(5) = (headb_2(3)**2 + headb_2(4)**2)**0.5
headb_2(11) = headb_2(11) + dbin_q
headb_2(12) = headb_2(12) + dtop_q
headb_2(13) = headb_2(13) + dbtm_q

time1 = time2
ens_num1 = ens_num2
heada_1 = heada_2
bdepth1 = bdepth2
```

```
      headb_1   headb_2

      depth1   depth2
      vel_mag1   vel_mag2
      vel_dirn1   vel_dirn2
      east_vel1   east_vel2
      north_vel1   north_vel2
      vert_vel1   vert_vel2
      error_vel1   error_vel2
      db_11   db_12
      db_21   db_22
      db_31   db_32
      db_41   db_42
      per_good1   per_good2
      discharge1   discharge2
      last_t   headb_1(2)
      ic2   0

      goto 35

   90 continue


      end

c-----------------------------------------------------------------------

      subroutine clearscreen
      do k   1, 25
        write(*, *)
      enddo
      end
```

APPENDIX D

GPSQ Version 1.0: User Manual

INTRODUCTION

This manual describes version 1 of a program designed to utilise data from an external Differential Global Positioning System to re-calculate river discharges measured using an RDI instruments ADCP.

The source code for the computer program is written in FORTRAN and is listed in Appendix B. An executable version of the code is contained on a MS-DOS formatted disk, labelled "GPSQ Ver 1.0". The program will run on most IBM compatible P.C.s. This program may be freely copied, but the source code may not be distributed to other parties. The author accepts no responsibility or liability resulting from the use of the program.

INSTALLATION

To install GPSQ ver 1.0 onto your hard drive, insert the diskette provided into the a: drive of your machine and type a:\install. A directory named GPSQ will then be created on your hard drive, the program files will then be copied into this directory.

RUNNING THE PROGRAM

Run the program by typing GPSQ at the command prompt in the GPSQ directory. A title screen should then be displayed, press ENTER to continue. The program then runs through the following stages:

(1) The user is prompted for the name of the ADCP and Navigation (GPS) data files, if the files are not in the current directory then a path must be specified. Sample data files; ADCP.DAT and GPS.DAT, are contained on the installation disk.

(2) The user is asked to input a name for the final discharge data output file. If no path is specified, the file is put placed in the current directory.

(3) A menu is displayed on the screen, giving options for data output units, choose from either English units (ft, ft/s, cfs) or SI units (m, m/s, cumecs).

(4) A second menu is displayed, asking the user to input a choice for the method to be used for top layer velocity estimation, either "power" or "constant".

(5) A 'depth calculation options' menu is displayed, either ADCP beam depths or FATHOMETER data can be chosen. At present the fathometer depth data option is not available, and if choosen an error message to this affect will appear.

(6) The program prompts the user for the ADCP beam angle (in degrees). If this information is unknown, it can be determined by pressing F9 in the RDI PLAYBACK program.

(7) The program asks for the local magnetic declination in degrees, if not already corrected for in the ADCP configuration file.

(8) A bottom track declination is then displayed on the screen. The bottom track declination is equal to the average directional difference between the bottom track data and GPS data. Hit ENTER to continue

(9) A power-law co-efficient is then required by the program, a value of 0.1667 is recommended for most applications, for more information see Chen, (1991).

(10) There is then a short delay whilst the program processes the velocity data. A menu is then displayed asking the user whether or not the require the near shore discharge calculation option.

(11) The final results of the discharge calculation are then displayed on the screen in terms of total discharge, total area, average velocity.

(12) During the program two files are created, a velocity data file and discharge data

file. The format of these files are described in the main body of this report.

APPENDIX E

ADCPFILL: User Manual

INTRODUCTION

This manual describes a program designed to fill in missing data from processed ADCP data files, in the RDI TRANSECT program format.

The source code for the computer program is written in FORTRAN and is listed in Appendix C. An executable version of the code is contained on a MS-DOS formatted disk, labelled "ADCPFILL". The program will run on most IBM compatible P.C.s. This program may be freely copied, but the source code may not be distributed to other parties. The author accepts no responsibility or liability resulting from the use of the program.

INSTALLATION

To install ADCPFILL onto your hard drive, insert the diskette provided into the a: drive of your machine and type a:\install. A directory named ADCPFILL will then be created on your hard drive, the program files will then be copied into this directory.

RUNNING THE PROGRAM

Run the program by typing ADCPFILL at the command prompt in the ADCPFILL directory. A title screen should then be displayed, press ENTER to continue. The program then runs through the following stages:

(1) The user is prompted for the name of the processed ADCP data files, if the files are not in the current directory then a path must be specified.

(2) The program then processes the ADCP file, checking for bad ensembles. The numbers of any bad ensembles are displayed on the screen at this time.

(3) A new ADCP data file, CLEAN.OUT, with all bad or missing data segments filled in is produced in the current working directory.